UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/847,534	05/01/2001	Lev Novik	MS1-694US	4018

22801 7590 12/14/2006

LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER
BULLOCK JR, LEWIS ALEXANDER

ART UNIT	PAPER NUMBER
2195	

DATE MAILED: 12/14/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/847,534	NOVIK ET AL.	
	Examiner	Art Unit	
	Lewis A. Bullock, Jr.	2195	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 25 September 2006.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-25,27-31,34,35 and 37-43* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-25, 27-31, 34, 35 and 37-43* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. In view of the appeal brief filed on September 25, 2006, PROSECUTION IS HEREBY REOPENED. The non-final rejection is set forth below.

To avoid abandonment of the application, appellant must exercise one of the following two options:

(1) file a reply under 37 CFR 1.111 (if this Office action is non-final) or a reply under 37 CFR 1.113 (if this Office action is final); or,

(2) initiate a new appeal by filing a notice of appeal under 37 CFR 41.31 followed by an appeal brief under 37 CFR 41.37. The previously paid notice of appeal fee and appeal brief fee can be applied to the new appeal. If, however, the appeal fees set forth in 37 CFR 41.20 have been increased since they were previously paid, then appellant must pay the difference between the increased fees and the amount previously paid.

A Supervisory Patent Examiner (SPE) has approved of reopening prosecution by signing below:

Applicant's arguments regarding the registration in relation to claim 1 is persuasive.

Claim Rejections - 35 USC § 101

2. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 28-31 and 34 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The cited claims detail an apparatus that does not correspond to one of the statutory category of inventions. The claims are not a method, and to be in the other cited categories requires at least one physical component or a combination of physical components, e.g. component of a computer system, a physical medium or combination of components. The claims allude to a software architecture which do not fit in one of the statutory category of inventions unless the architecture is executing on a computer system or stored on a storage medium.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-25, 27-31, 34, 35 and 37-43 are rejected under 35 U.S.C. 103(a) as being unpatentable over DU (U.S. Patent 6,041,306) in view of FERIDUM (U.S. Patent 6,336,139).

As to claims 1 and 10, DU teaches a computer-implemented method / system comprising: a function implemented as a state machine (rule node instance state machine) that receives a plurality of events and identifies an event to which an update consumer (rule) has subscribed, wherein the update consumer (rule) is associated with the state machine (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status…(e) Raises events..") (col. 20, lines 1-32); applying the update consumer (rule) to the state machine (rule node instance state machine) in response to the identified event (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status…(e) Raises events..") (col. 20, lines 1-32); and generating a specific event if the function is satisfied by the plurality of events (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status…(e) Raises events..") (col. 20, lines 1-32). However, DU does not explicitly mention that the function performed by the state machine to the events is correlation.

FERIDUN teaches a computer-implemented method / medium of correlating events comprising instructions for: receiving a plurality of events (events from event streams); applying the plurality of events to a correlation function (correlation rule), wherein the correlation function (correlation rule) is implemented as a state machine (state machines) and is configured to correlate the plurality of events (events); and generating a specific event (action / control signal / another event) if the correlation function is satisfied by the plurality of events (col. 2, lines 43-62; col. 3, lines 9-20; col.

9, lines 15-22; col. 9, lines 41-57; col. 10, lines 5-19; col. 12, lines 32-47). It would be obvious to one of ordinary skill in the art that since the rule instance state machine of DU also has transitions and states that are manipulated by received events as detailed in DU that the state machine of DU is the same state machine of FERIDUN. Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of DU with the teachings of FERIDUN in order to implement a set of simple or low-level correlation rules, each of which may be useful in recognizing a given pattern of one or more events indicative of a given condition sought to be monitored and/or controlled (col. 2, lines 4-9).

As to claims 11 and 19, DU teaches a computer-implemented method / system comprising: a plurality of functions (rule nodes) (col. 7, lines 5-16) wherein each function is implemented as a state machine (rule node instance state machine) that receives a plurality of events and data elements (run-time information / process-relevant data that specifies rules) and is associated with an update consumer (rule) that updates the state of the associated state machine (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status…(e) Raises events..") (col. 20, lines 1-32; see also col. 11, lines 52-64); applying the events and data elements to the correlation functions (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status...(e) Raises events..") (col. 20, lines 1-32); and generating a specific event if the function is satisfied (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status...(e) Raises events..") (col. 20, lines 1-32). However, DU does not explicitly mention that the function performed by the state machine to the events is correlation.

FERIDUN teaches a computer-implemented method embodied in a computer-readable medium that is executable by a processor comprising instructions for: receiving a plurality of events (events from one event stream); receiving a plurality of data elements (information from a subsequent event stream / data stream); identifying a plurality of correlation functions (correlators of the plurality of agents) configured to correlate the plurality of events and the plurality of data elements wherein each correlation function is implemented with an associated state machine (via the registered interest in events by agents); applying the plurality of events and the plurality of data elements to the plurality of correlation functions (via sending events to agents and correlating the streams); and generating a specific event if at least one of the plurality of correlation functions is satisfied (generation of an event issued to another node) (col. 9, lines 41-57; col. 12, lines 62-67; col. 10, lines 5-19; col. 6, lines 21-60; col. 2, lines 38-

62; col. 8, lines 15-28). It would be obvious to one of ordinary skill in the art that since the rule instance state machine of DU also has transitions and states that are manipulated by received events as detailed in DU that the state machine of DU is the same state machine of FERIDUN. Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of DU with the teachings of FERIDUN in order to implement a set of simple or low-level correlation rules, each of which may be useful in recognizing a given pattern of one or more events indicative of a given condition sought to be monitored and/or controlled (col. 2, lines 4-9).

As to claims 20 and 27, DU teaches a computer-implemented method / system comprising: creating an instance of a particular state machine (rule node instance state machine) and defining transitions (arcs / transitions) for the state machine by subscribing to at least one event (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status...(e) Raises events..") (col. 20, lines 1-32); and applying an update consumer to the state machine to update the state of the state machine (rule node instance state machine) in response to the identified event (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status...(e) Raises events..") (col. 20, lines 1-32). It would be inherent that since a state machine is created into memory that it has a schema that defines it and which must be identified in order to create that state machine. However, DU does not explicitly mention that neither the function performed by the state machine to the events is correlation nor that the update consumer is a class object. DU does teach that the rule is implemented in a rule language (col. 11, lines 52-65). Official Notice is taken in that it is well known in the art that a rule language is an object oriented language wherein the rules implemented in that language would obviously be in classes (see publications "A Rule Engine for Query Tranformation in Starburst and IBM DB2 C/S DBMS" by Pirahesh et al; "Rule-based systems formalized within a software architectural style" by GAMBEL et al; "Integrating General Knowledge with Object-Oriented Case Representation and Reasoning" by Bergmann et al; or "On the Implementation of Finite State Machines" by Van Gurp et al.). Therefore, it would be obvious to one of ordinary skill in the art that the update consumer (rules) is in a class since a rule language is an object-oriented representation.

FERIDUN teaches a computer-implemented method embodied in a computer-readable medium that is executable by a processor comprising instructions for: having the state machines (correlators having correlation rules) to correlate at least two events (events / event streams) (col. 2, lines 43-62; col. 3, lines 9-20; col. 9, lines 15-22; col. 9, lines 41-57; col. 10, lines 5-19; col. 12, lines 32-47); creating an instance of a particular state machine (via breeder routine / non-mobile event processing host instantiating agents that have correlation rules) (col. 10, lines 47-58); and defining transitions for the particular state machine (col. 8, lines 15-28). FERIDUN also teaches that the agents / state machines are implemented in Java (col. 6, lines 56-60). It would be obvious to one of ordinary skill in the art that since the rule instance state machine of DU also has transitions and states that are manipulated by received events as detailed in DU that the state machine of DU is the same state machine of FERIDUN. Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of DU with the teachings of FERIDUN in order to implement a set of simple or low-level correlation rules, each of which may be useful in recognizing a given pattern of one or more events indicative of a given condition sought to be monitored and/or controlled (col. 2, lines 4-9).

As to claim 28, DU teaches a computer-implemented method / system comprising: a plurality of functions (rule nodes) (col. 7, lines 5-16) wherein each function is implemented as a state machine (rule node instance state machine) that receives a plurality of events and data elements (run-time information / process-relevant data that specifies rules) and is associated with an update consumer (rule) that updates the state of the associated state machine (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status...(e) Raises events..") (col. 20, lines 1-32; see also col. 11, lines 52-64); applying the events and data elements to the correlation functions (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status…(e) Raises events..") (col. 20, lines 1-32); and generating a specific event if the function is satisfied (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status…(e) Raises events..") (col. 20, lines 1-32). However, DU does not explicitly mention that the function performed by the state machine to the events is correlation.

FERIDUN teaches an apparatus comprising: a plurality of event consumers (registered agents); and an event correlator (correlation rule / correlator) coupled to the plurality of event consumers (col. 9, lines 1-9; fig. 6), the event correlator to receive events from at least one event source (events received from one event stream) and to receive data elements (information received from another event stream / data stream) from at least one data source, the event correlator further to receive at least one correlation function (correlation rules) configured to correlate events and data elements and to apply the received events and the received data elements to the correlation function, wherein the correlation function is implemented by a state machine (via sending events to agents and correlating the streams), wherein the event correlator generates a specific event if the received events and the received data satisfy the correlation function (generation of an event issued to another node) (col. 9, lines 41-57; col. 12, lines 62-67; col. 10, lines 5-19; col. 6, lines 21-60; col. 2, lines 38-62; col. 8, lines 15-28). It would be obvious to one of ordinary skill in the art that since the rule instance state machine of DU also has transitions and states that are manipulated by received events as detailed in DU that the state machine of DU is the same state machine of FERIDUN. Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of DU with the teachings of FERIDUN in order to implement a set of simple or low-level correlation rules, each of which may be useful in recognizing a given pattern of one or more events indicative of a given condition sought to be monitored and/or controlled (col. 2, lines 4-9).

As to claim 35, DU teaches a computer-implemented method / system comprising: a plurality of functions wherein each function is implemented as a state machine (rule node instance state machine) that receives a plurality of events and identifies an event to which an update consumer (rule) has subscribed, wherein the update consumer (rule) is associated with the state machine (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status...(e) Raises events..") (col. 20, lines 1-32); applying the plurality of events to the plurality of functions to determine whether any of the functions are satisfied wherein the plurality of events are applied by causing update consumers (rules) to the state machine (rule node instance state machine) in response to the identified event to update the state of the associated state machine (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status…(e) Raises events..") (col. 20, lines 1-32); and generating a specific event if the function is satisfied by the plurality of events (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status…(e) Raises events..") (col. 20, lines 1-32). However, DU does not explicitly mention that the function performed by the state machine to the events is correlation.

FERIDUN teaches one or more computer readable media having stored thereon a computer program that, when executed by one or more processors, causes the one or more processors to: receive a plurality of events (events from event streams); identify a plurality of correlation functions (correlation rules) configured to correlate the plurality of events wherein each of the plurality of correlation functions is implemented as a state machine; apply the plurality of events to the plurality of correlation functions to determine whether any of the plurality of correlation functions are satisfied by the plurality of events; and generate a specific event (action / control signal / another event) if one of the plurality of correlation functions is satisfied by the plurality of events (col. 2, lines 43-62; col. 3, lines 9-20; col. 9, lines 15-22; col. 9, lines 41-57; col. 10, lines 5-19; col. 12, lines 32-47). It would be obvious to one of ordinary skill in the art that since the rule instance state machine of DU also has transitions and states that are manipulated by received events as detailed in DU that the state machine of DU is the same state machine of FERIDUN. Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of DU with the teachings of FERIDUN in order to implement a set of simple or low-level correlation rules, each of which may be useful in recognizing a given pattern of one or more events indicative of a given condition sought to be monitored and/or controlled (col. 2, lines 4-9).

As to claim 40, DU teaches a computer-implemented method / system comprising: a plurality of functions wherein each function is implemented as a state machine (rule node instance state machine) that receives a plurality of events and identifies an event to which an update consumer (rule) has subscribed, wherein the update consumer (rule) is associated with a particular state machine to update the state of the state machine (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status...(e) Raises events..") (col. 20, lines 1-32; see also col. 11, lines 52-64); applying the plurality of events to the plurality of functions to determine whether any of the functions are satisfied wherein the plurality of events are applied by causing update consumers (rules) to the state machine (rule node instance state machine) in response to the identified event to update the state of the associated state machine (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status…(e) Raises events..") (col. 20, lines 1-32); and generating a specific event if the function is satisfied by the plurality of events (rules defined in the rule node will then be evaluated by the OpenPM engine. Rules in the rule node are also evaluated when one of the events to which it subscribes occurs. Generally a rule, if evaluated to be TRUE, does one of the following: (a) Fires one or more of its outward arcs; (b) Changes the rule node instance status…(e) Raises events..") (col. 20, lines 1-32). It would be inherent to the teachings of DU that since each rule node receives events and is associated with certain registered events, the rule nodes must be created and are associated with certain event types. However, DU does not explicitly mention that the function performed by the state machine to the events is correlation such that if the events are correlated an additional event is sent to the event consumer.

FERIDUN teaches a method comprising: receiving events from event providers (events from event streams); creating a first state machine (initial state machine); creating a second state machine (subsequent state machine); associating a first event type with the first state machine (event is handled by the first state machine);

associating a second event type with the second state machine (event is handled by second state machine); in response to receiving an event having a first event type, performing an action to the first state machine (if event is handled by initial state machine perform action); in response to receiving an event having a second event type, applying an action to the second state machine (if event is handled by subsequent state machine perform action) (col. 9, lines 58-65; col. 3, lines 21-27; col. 11, lines 11-23; col 12, lines 32-61), and if the events are correlated, generating an additional event (action / control signal / another event); and sending the additional event to an event consumer (another node) (col. 2, lines 43-62; col. 3, lines 9-20; col. 9, lines 15-22; col. 9, lines 41-57; col. 10, lines 5-19; col. 12, lines 32-54). It would be obvious to one of ordinary skill in the art that since the rule instance state machine of DU also has transitions and states that are manipulated by received events as detailed in DU that the state machine of DU is the same state machine of FERIDUN. Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of DU with the teachings of FERIDUN in order to implement a set of simple or low-level correlation rules, each of which may be useful in recognizing a given pattern of one or more events indicative of a given condition sought to be monitored and/or controlled (col. 2, lines 4-9).

As to claim 2, neither DU nor FERIDUN explicitly mention that the update consumer is a class object. DU does teach that the rule is implemented in a rule language (col. 11, lines 52-65). Official Notice is taken in that it is well known in the art that a rule language is an object oriented language wherein the rules implemented in that language would obviously be in classes (see publications “A Rule Engine for Query Tranformation in Starburst and IBM DB2 C/S DBMS” by Pirahesh et al; “Rule-based systems formalized within a software architectural style” by GAMBEL et al; “Integrating General Knowledge with Object-Oriented Case Representation and Reasoning” by Bergmann et al; or “On the Implementation of Finite State Machines” by Van Gurp et al.). Therefore, it would be obvious to one of ordinary skill in the art that the update consumer (rules) is in a class since a rule language is an object-oriented representation.

As to claims 3 and 4, FERIDUN teaches receiving a data element (s) (information from a subsequent event stream / data stream); and applying the data element (s) and at least one of the plurality of events to the correlation function (via sending events to agents and correlating the streams) (col. 9, lines 41-57; col. 12, lines 62-67; col. 10, lines 5-19; col. 6, lines 21-60; col. 2, lines 38-62; col. 8, lines 15-28).

As to claim 5, FERIDUN teaches communicating the specific event (action / control signal / another event) to at least one event consumer (registered agent) that subscribed to the specific event (col. 2, lines 43-62; col. 3, lines 9-20; col. 9, lines 15-22; col. 9, lines 41-57; col. 10, lines 5-19; col. 12, lines 32-47).

As to claim 6, FERIDUN teaches continuing to receive additional events (events) and apply the additional events (events) to the correlation function (correlation rules) if the correlation function is not satisfied by the plurality of events (col. 8, lines 15-34; col. 8, lines 53-61).

As to claim 7, FERIDUN teaches resetting the correlation function after generating a specific event (via reset rules / after handling of event, correlation rule de-activates itself) (col. 9, lines 36-37; col. 9, lines 54-57; col. 12, lines 32-45).

As to claim 8, FERIDUN teaches creating an instance of a particular state machine (via breeder routine / non-mobile event processing host instantiating agents that have correlation rules) (col. 10, lines 47-58); and defining transitions for the particular state machine by subscribing to at least one event (via registering interests in events) (col. 8, lines 15-28).

As to claim 9, DU and FERIDUN substantially disclose the invention as disclosed above. In addition, DU and FERIDUN teach de-activating the correlation function, i.e. state machine, when either its criteria have been met or a specified timeout has occurred (col. 12, lines 32-40). However, DU and FERIDUN do not explicitly teach deleting or removing a state machine once it has reached a final state. Official Notice is taken in that it is well known in the art that a state machine is removed or deleted once it has reached a final or target state. See U.S. Patents 5930,482; 5,913,043; 5,958,035; 6,307,546; and U.S. Patent Publication 2002/0040409 A1 for examples. Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention use the well known technique deleting the state machine once it has reached a final state with the system of DU and FERIDUN in order to facilitate the manage the operation of state machines based on state changes.

As to claim 12, refer to claim 9 for rejection.

As to claim 13, refer to claim 2 for rejection.

As to claim 14, refer to claim 5 for rejection.

As to claims 15 and 16, refer to claim 3 and 4 for rejection. However, claim 16 further details receiving additional correlation functions. FERIDUN teaches the registering of agents via the registering of correlation functions (col. 8, lines 23-27) and the dynamic generation of agents (col. 9, lines 54-57). It is inherent within the teachings of FERIDUN that when agents are generated they register correlation functions in order to register interest in particular events, thereby teaching the step of receiving additional correlation functions.

As to claim 17, FERIDUN teaches the registering of correlation functions (col. 8, lines 23-27) and generating a specific event if a correlation function is satisfied (col. 2, lines 43-62; col. 3, lines 9-20; col. 9, lines 15-22; col. 9, lines 41-57; col. 10, lines 5-19; col. 12, lines 32-47). It is inherent within the teachings of FERIDUN that the additional registered correlation functions of the generated agent also generate a specific event when satisfied.

As to claim 18, FERIDUN teaches the specific event generated is dependent on which correlation function is satisfied (depending on the rule requirements in order for an event to be satisfied, i.e. if a specific number of the same type of event has occurred, a threshold rule is triggered, whether an event matches a search criteria, a matching rule is triggered) (col. 9, lines 15-40).

As to claims 21 and 22, DU and FERIDUN substantially disclose the invention as disclosed above. In addition, DU and FERIDUN teach de-activating the correlation function, i.e. state machine, when either its criteria have been met or a specified timeout has occurred (col. 12, lines 32-40). However, DU and FERIDUN do not explicitly teach deleting or removing a state machine once it has reached a final state. Official Notice is taken in that it is well known in the art that a state machine is removed or deleted once it has reached a final or target state. See U.S. Patents 5930,482; 5,913,043; 5,958,035; 6,307,546; and U.S. Patent Publication 2002/0040409 A1 for examples. Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention use the well known technique deleting the state machine once it has reached a final state with the system of DU and FERIDUN in order to facilitate the manage the operation of state machines based on state changes.

As to claims 23 and 24, FERIDUN teaches the particular state machine correlates at least one event (events from one event stream) and at least one data element (information from a subsequent event stream / data stream) (col. 9, lines 41-57; col. 12, lines 62-67; col. 10, lines 5-19; col. 6, lines 21-60; col. 2, lines 38-62; col. 8, lines 15-28).

As to claim 25, FERIDUN teaches determining a current state of the particular state machine (whether the state machine is satisfied / whether the correlation rule is either active or de-active) (col. 11, lines 11-23; col. 11, line 61 – col. 12, line 9).

As to claims 29-31, FERIDUN teaches the event correlator communicates the specific event (event) to event consumers that have requested to receive the specific event (agents registered to receive the event) by using a filter (intermediate location / monitor) of the consumers (col. 8, lines 15-28 col. 8, lines 53-61; col. 8, lines 62-67).

As to claim 34, FERIDUN teaches continuing to receive additional events (events) and additional data elements (other events / data from data stream) and apply the additional events and data elements (events / data from data stream) to the correlation function (correlation rules) (col. 8, lines 15-34; col. 8, lines 53-61; col. 12, lines 62-67).

As to claim 37, neither DU nor FERIDUN explicitly mention that the update consumer is a class object. DU does teach that the rule is implemented in a rule language (col. 11, lines 52-65). Official Notice is taken in that it is well known in the art that a rule language is an object oriented language wherein the rules implemented in that language would obviously be in classes (see publications "A Rule Engine for Query Tranformation in Starburst and IBM DB2 C/S DBMS" by Pirahesh et al; "Rule-based systems formalized within a software architectural style" by GAMBEL et al; "Integrating General Knowledge with Object-Oriented Case Representation and Reasoning" by Bergmann et al; or "On the Implementation of Finite State Machines" by Van Gurp et al.). Therefore, it would be obvious to one of ordinary skill in the art that the update consumer (rules) is in a class since a rule language is an object-oriented representation.

As to claim 38, FERIDUN teaches identify a current state of the state machine (whether rule is active or de-active) (col. 11, lines 11-23; col. 11, line 61 – col. 12, line 9).

As to claim 39, FERIDUN teaches the state machines (correlators having correlation rules) to correlate at least two events (events / event streams) (col. 2, lines 43-62; col. 3, lines 9-20; col. 9, lines 15-22; col. 9, lines 41-57; col. 10, lines 5-19; col. 12, lines 32-47); creating an instance of a particular state machine (via breeder routine / non-mobile event processing host instantiating agents that have correlation rules) (col.

10, lines 47-58); and defining transitions for the particular state machine by subscribing to at least one event (via registering interests in events) (col. 8, lines 15-28).

As to claim 41, DU and FERIDUN substantially disclose the invention as disclosed above. In addition, DU and FERIDUN teach de-activating the correlation function, i.e. state machine, when either its criteria have been met or a specified timeout has occurred (col. 12, lines 32-40). However, DU and FERIDUN do not explicitly teach deleting or removing a state machine once it has reached a final state. Official Notice is taken in that it is well known in the art that a state machine is removed or deleted once it has reached a final or target state. See U.S. Patents 5930,482; 5,913,043; 5,958,035; 6,307,546; and U.S. Patent Publication 2002/0040409 A1 for examples. Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention use the well known technique deleting the state machine once it has reached a final state with the system of DU and FERIDUN in order to facilitate the manage the operation of state machines based on state changes.

As to claims 42 and 43, FERIDUN teaches the additional event is sent to the event consumer through a filter (intermediate location / monitor) associated with the event logging consumer (subscribed consumer) (col. 8, lines 15-28; col. 8, lines 53-61; col. 8, lines 62-67; col. 9, lines 54-57; col. 12, lines 48-54).

Response to Arguments

5. Applicant's arguments with respect to claims 1-25, 27-31, 34, 35, and 37-43 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (571) 272-3759. The examiner can normally be reached on Monday-Friday, 8:30 a.m. - 5:00 p.m..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a

USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

December 8, 2006

[illegible]

LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER

[illegible]

MENG-AL T. AN
SUPERVISORY PATENT EXAMINER
[illegible]OLOGY CENTER 2100